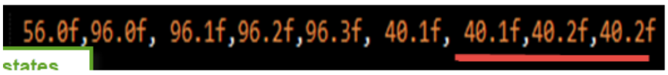
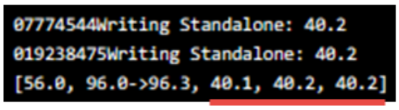
THESE ARE ERRORS:



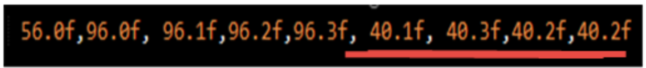
```
56.0f,96.0f, 96.1f,96.2f,96.3f, 40.1f, 40.1f,40.2f,40.2f
```



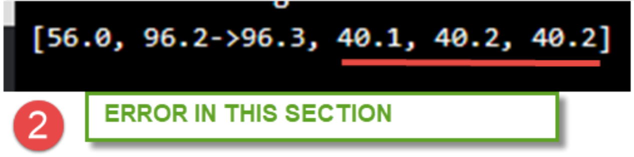
```
07774544Writing Standalone: 40.2
019238475Writing Standalone: 40.2
[56.0, 96.0->96.3, 40.1, 40.2, 40.2]
```

1

ERROR IN THIS SECTION



```
56.0f,96.0f, 96.1f,96.2f,96.3f, 40.1f, 40.3f,40.2f,40.2f
```

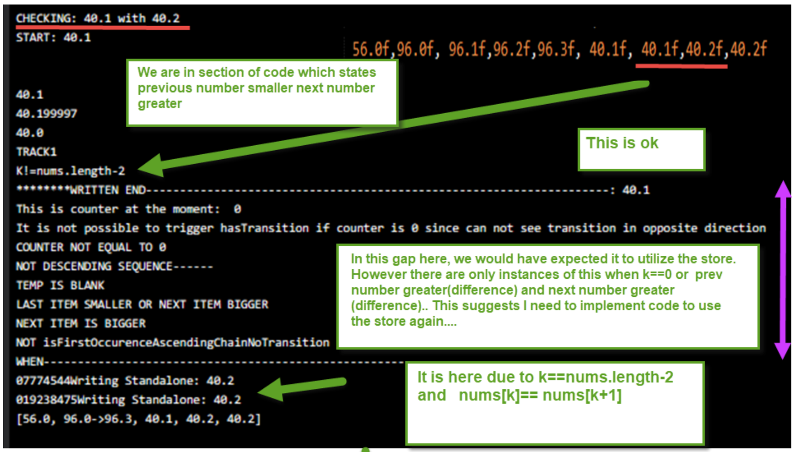


```
[56.0, 96.2->96.3, 40.1, 40.2, 40.2]
```

2

ERROR IN THIS SECTION

FIXING ERROR 1:



```
CHECKING: 40.1 with 40.2
START: 40.1


40.1
40.199997
40.0
TRACK1
K!=nums.length-2
*******WRITTEN END---------------------------------------------------------------------: 40.1
This is counter at the moment:  0
It is not possible to trigger hasTransition if counter is 0 since can not see transition in opposite direction
COUNTER NOT EQUAL TO 0
NOT DESCENDING SEQUENCE------
TEMP IS BLANK
LAST ITEM SMALLER OR NEXT ITEM BIGGER
NEXT ITEM IS BIGGER
NOT isFirstOccurenceAscendingChainNoTransition
WHEN-------------------------------------------------------
07774544Writing Standalone: 40.2
019238475Writing Standalone: 40.2
[56.0, 96.0->96.3, 40.1, 40.2, 40.2]
```

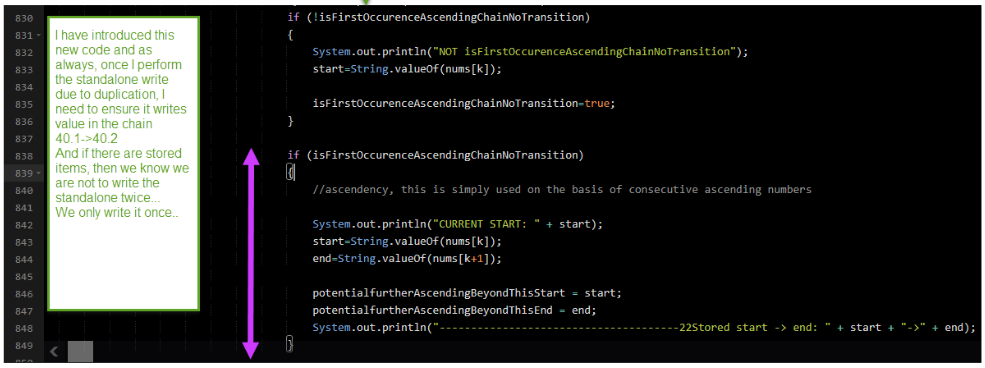
```
56.0f,96.0f, 96.1f,96.2f,96.3f, 40.1f, 40.1f,40.2f,40.2f
```

We are in section of code which states previous number smaller next number greater

This is ok

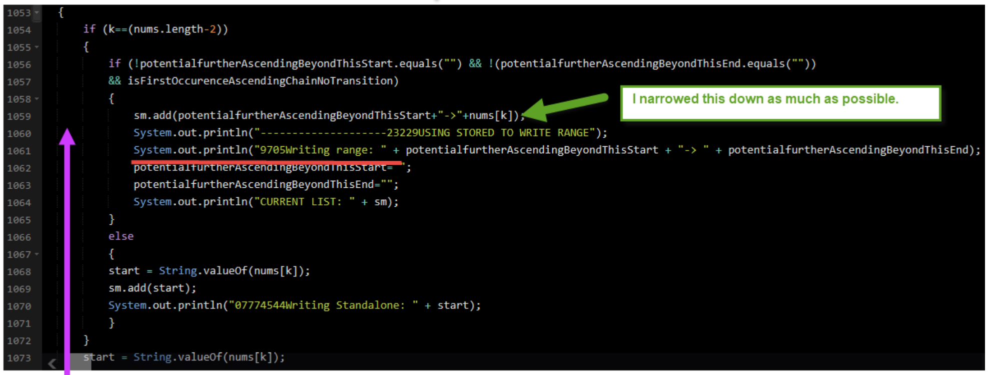
In this gap here, we would have expected it to utilize the store. However there are only instances of this when k==0 or prev number greater(difference) and next number greater (difference).. This suggests I need to implement code to use the store again....

It is here due to k==nums.length-2 and nums[k]== nums[k+1]



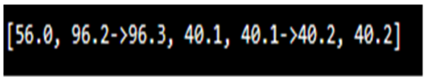
I have introduced this new code and as always, once I perform the standalone write due to duplication, I need to ensure it writes value in the chain 40.1->40.2
And if there are stored items, then we know we are not to write the standalone twice...
We only write it once..

```java
if (!isFirstOccurenceAscendingChainNoTransition)
{
    System.out.println("NOT isFirstOccurenceAscendingChainNoTransition");
    start=String.valueOf(nums[k]);

    isFirstOccurenceAscendingChainNoTransition=true;
}

if (isFirstOccurenceAscendingChainNoTransition)
{
    //ascendency, this is simply used on the basis of consecutive ascending numbers

    System.out.println("CURRENT START: " + start);
    start=String.valueOf(nums[k]);
    end=String.valueOf(nums[k+1]);

    potentialfurtherAscendingBeyondThisStart = start;
    potentialfurtherAscendingBeyondThisEnd = end;
    System.out.println("------------------------------------22Stored start -> end: " + start + "->" + end);
}
```



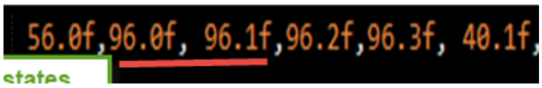
```java
{
    if (k==(nums.length-2))
    {
        if (!potentialfurtherAscendingBeyondThisStart.equals("") && !(potentialfurtherAscendingBeyondThisEnd.equals(""))
        && isFirstOccurenceAscendingChainNoTransition)
        {
            sm.add(potentialfurtherAscendingBeyondThisStart+"->"+nums[k]);
            System.out.println("-------------------23229USING STORED TO WRITE RANGE");
            System.out.println("9705Writing range: " + potentialfurtherAscendingBeyondThisStart + "-> " + potentialfurtherAscendingBeyondThisEnd);
            potentialfurtherAscendingBeyondThisStart="";
            potentialfurtherAscendingBeyondThisEnd="";
            System.out.println("CURRENT LIST: " + sm);
        }
        else
        {
        start = String.valueOf(nums[k]);
        sm.add(start);
        System.out.println("07774544Writing Standalone: " + start);
        }
    }
    start = String.valueOf(nums[k]);
```

I narrowed this down as much as possible.



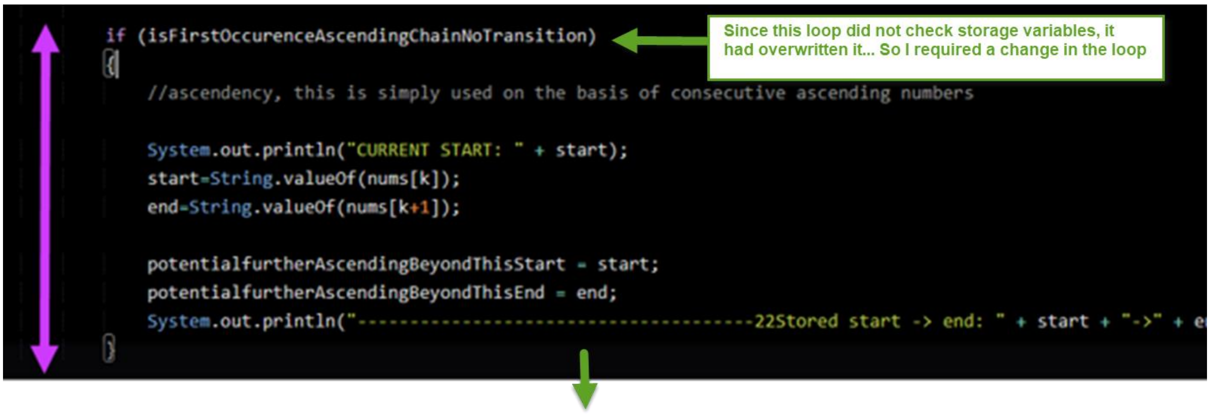
```
[56.0, 96.2->96.3, 40.1, 40.1->40.2, 40.2]
```

Infact we can see that it has cut off. This is because it has been overwritten by 96.2->96.3 since the above loop



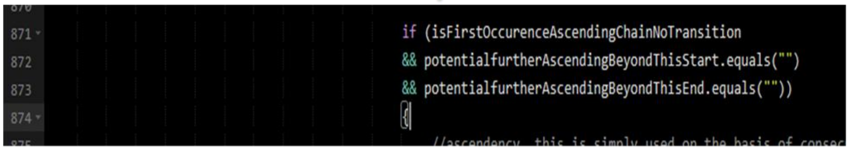
```
56.0f,96.0f, 96.1f,96.2f,96.3f, 40.1f,
```

states

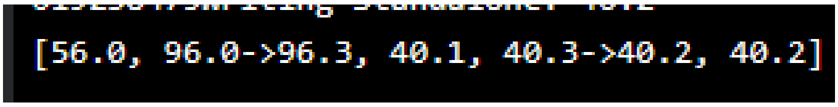


```
if (isFirstOccurenceAscendingChainNoTransition)
{
    //ascendency, this is simply used on the basis of consecutive ascending numbers

    System.out.println("CURRENT START: " + start);
    start=String.valueOf(nums[k]);
    end=String.valueOf(nums[k+1]);

    potentialfurtherAscendingBeyondThisStart = start;
    potentialfurtherAscendingBeyondThisEnd = end;
    System.out.println("-------------------------------------22Stored start -> end: " + start + "->" + e
}
```

Since this loop did not check storage variables, it had overwritten it... So I required a change in the loop

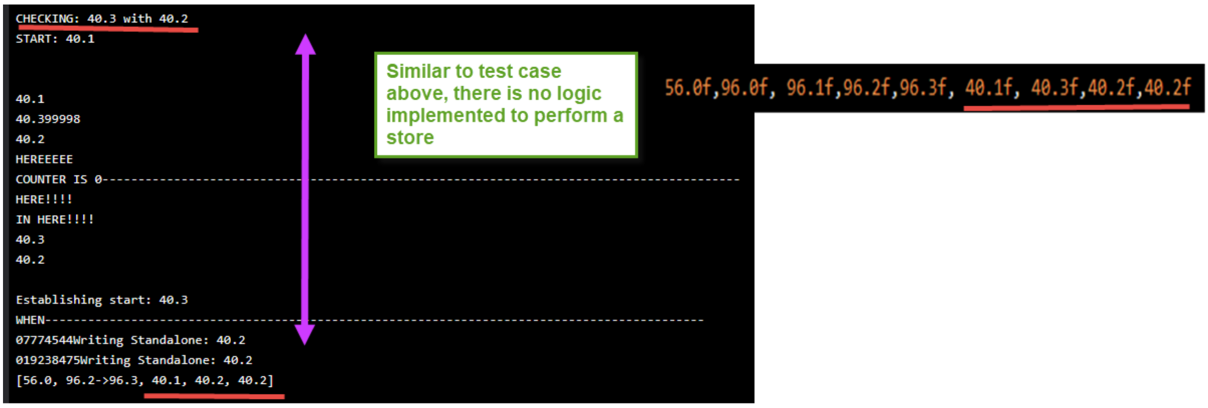


```
if (isFirstOccurenceAscendingChainNoTransition
&& potentialfurtherAscendingBeyondThisStart.equals("")
&& potentialfurtherAscendingBeyondThisEnd.equals(""))
{
    //ascendency, this is simply used on the basis of consec
```



```
[56.0, 96.0->96.3, 40.1, 40.3->40.2, 40.2]
```

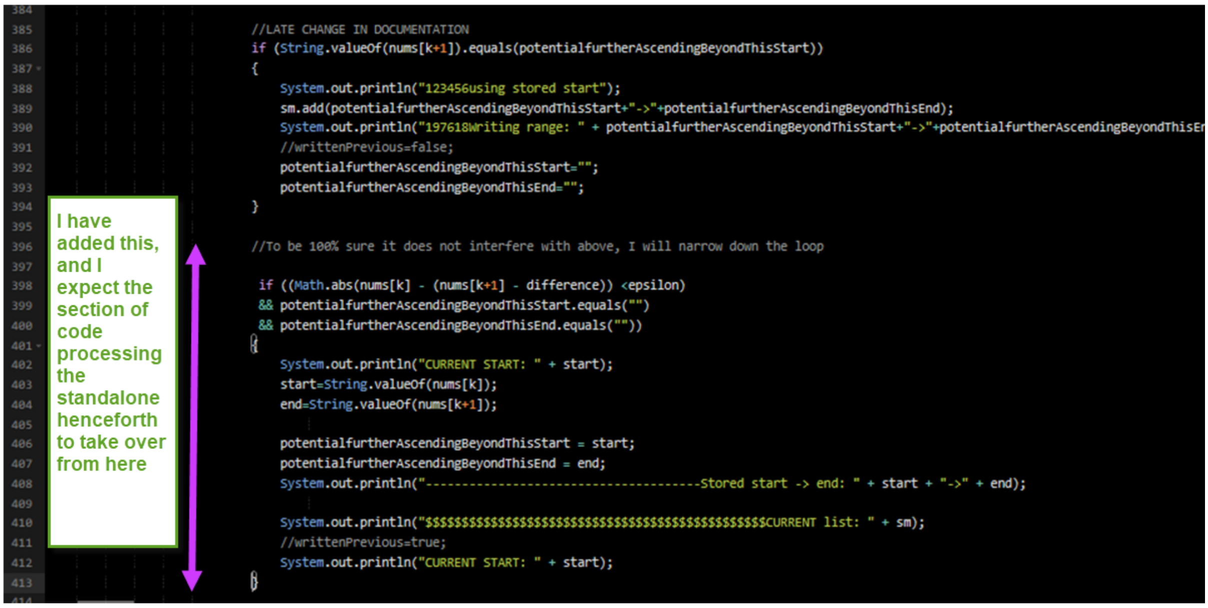
FIXING ERROR 2:



```
CHECKING: 40.3 with 40.2
START: 40.1


40.1
40.399998
40.2
HEREEEEE
COUNTER IS 0-----------------------------------------------------------------------------------------
HERE!!!!
IN HERE!!!!
40.3
40.2

Establishing start: 40.3
WHEN------------------------------------------------------------------------------------------------
07774544Writing Standalone: 40.2
019238475Writing Standalone: 40.2
[56.0, 96.2->96.3, 40.1, 40.2, 40.2]
```

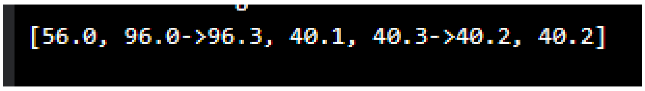
Similar to test case above, there is no logic implemented to perform a store

```
56.0f,96.0f, 96.1f,96.2f,96.3f, 40.1f, 40.3f,40.2f,40.2f
```



```
//LATE CHANGE IN DOCUMENTATION
if (String.valueOf(nums[k+1]).equals(potentialfurtherAscendingBeyondThisStart))
{
    System.out.println("123456using stored start");
    sm.add(potentialfurtherAscendingBeyondThisStart+"->"+potentialfurtherAscendingBeyondThisEnd);
    System.out.println("197618Writing range: " + potentialfurtherAscendingBeyondThisStart+"->"+potentialfurtherAscendingBeyondThisEn
    //writtenPrevious=false;
    potentialfurtherAscendingBeyondThisStart="";
    potentialfurtherAscendingBeyondThisEnd="";
}

//To be 100% sure it does not interfere with above, I will narrow down the loop

 if ((Math.abs(nums[k] - (nums[k+1] - difference)) <epsilon)
 && potentialfurtherAscendingBeyondThisStart.equals("")
 && potentialfurtherAscendingBeyondThisEnd.equals(""))
{
    System.out.println("CURRENT START: " + start);
    start=String.valueOf(nums[k]);
    end=String.valueOf(nums[k+1]);

    potentialfurtherAscendingBeyondThisStart = start;
    potentialfurtherAscendingBeyondThisEnd = end;
    System.out.println("------------------------------------Stored start -> end: " + start + "->" + end);

    System.out.println("$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$CURRENT list: " + sm);
    //writtenPrevious=true;
    System.out.println("CURRENT START: " + start);
}
```

I have added this, and I expect the section of code processing the standalone henceforth to take over from here

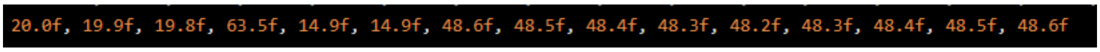


```
[56.0, 96.0->96.3, 40.1, 40.3->40.2, 40.2]
```

Like always, I will need to revisit all my test cases again....

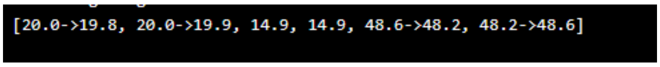
And it has failed against ChatGPT data in a way that has never appeared before..

So I am taking a small extract to investigate:

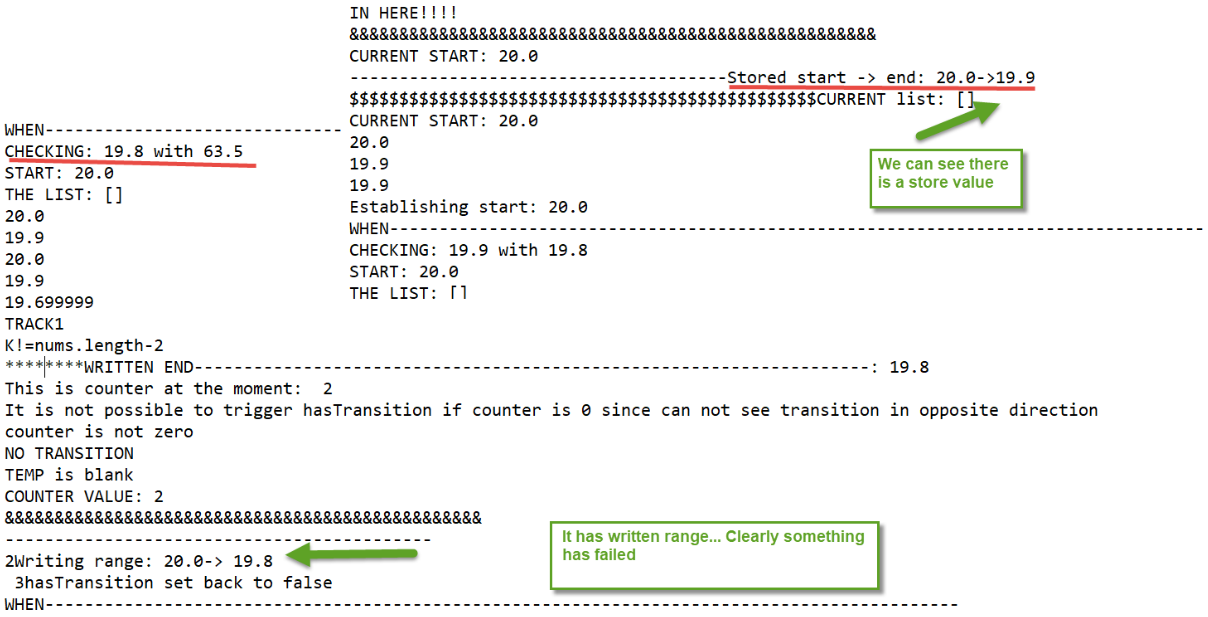


```
20.0f, 19.9f, 19.8f, 63.5f, 14.9f, 14.9f, 48.6f, 48.5f, 48.4f, 48.3f, 48.2f, 48.3f, 48.4f, 48.5f, 48.6f
```

We can see it has missed out 63.5f, we can see that it has created 20->19.9 which is irrelevant...



```
[20.0->19.8, 20.0->19.9, 14.9, 14.9, 48.6->48.2, 48.2->48.6]
```

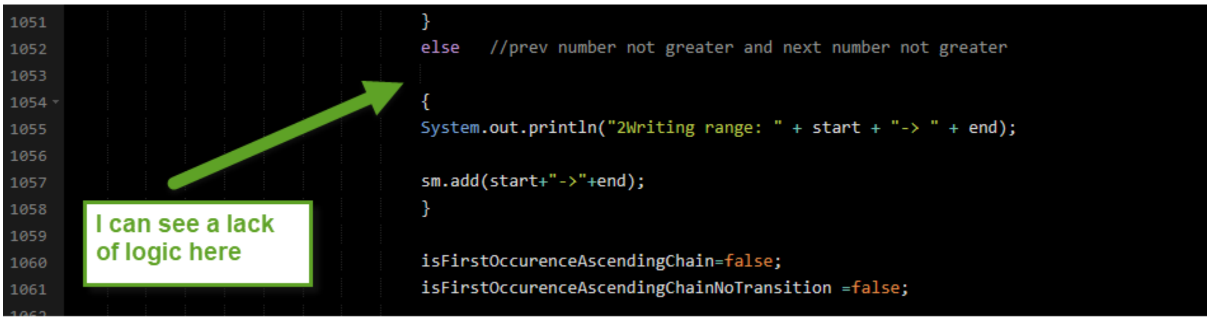


```
IN HERE!!!!
&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&
CURRENT START: 20.0
--------------------------------------Stored start -> end: 20.0->19.9
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$CURRENT list: []
CURRENT START: 20.0
20.0
19.9
19.9
Establishing start: 20.0
WHEN------------------------------------------------------------------------------------------
CHECKING: 19.9 with 19.8
START: 20.0
THE LIST: []
```

We can see there is a store value

```
WHEN-----------------------------
CHECKING: 19.8 with 63.5
START: 20.0
THE LIST: []
20.0
19.9
20.0
19.9
19.699999
TRACK1
K!=nums.length-2
********WRITTEN END------------------------------------------------------------------: 19.8
This is counter at the moment:  2
It is not possible to trigger hasTransition if counter is 0 since can not see transition in opposite direction
counter is not zero
NO TRANSITION
TEMP is blank
COUNTER VALUE: 2
&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&
-------------------------------------------
2Writing range: 20.0-> 19.8
 3hasTransition set back to false
WHEN-----------------------------------------------------------------------------------------
```

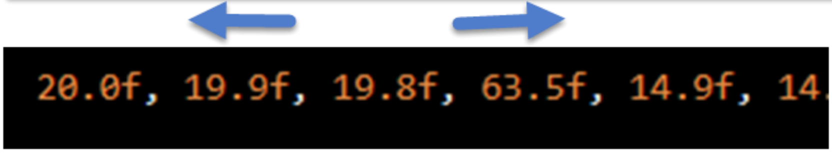
It has written range... Clearly something has failed



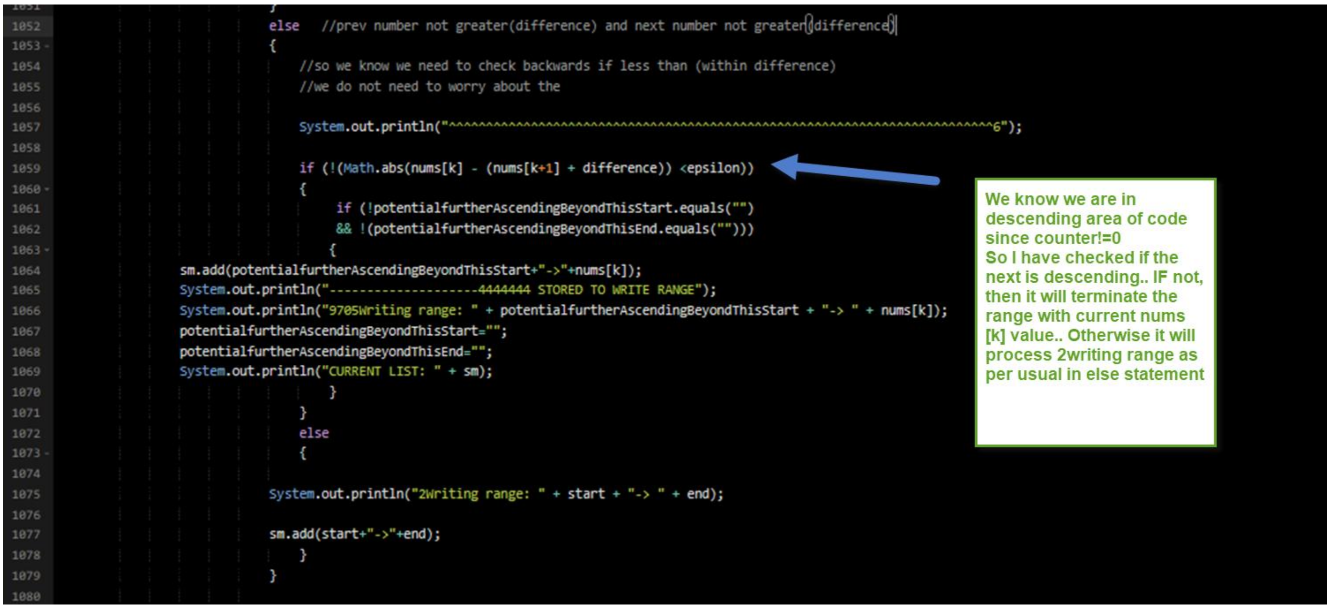
```
}
else    //prev number not greater and next number not greater

{
System.out.println("2Writing range: " + start + "-> " + end);

sm.add(start+"->"+end);
}

isFirstOccurenceAscendingChain=false;
isFirstOccurenceAscendingChainNoTransition =false;
```

I can see a lack of logic here

And we know that condition is correct. So need to use the store 20.0->19.9...
We check 19.8 with 63.5
Since it is not within difference, we used start of the Store and nums[k] as the end.
Then we can let it perform the next check...... 63.5 with 14.9

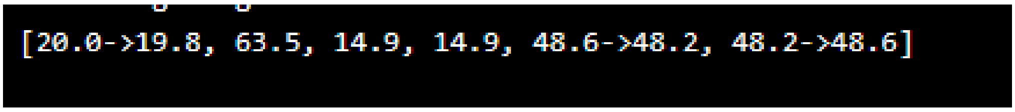


```
20.0f, 19.9f, 19.8f, 63.5f, 14.9f, 14.
```



```
                    }
                    else   //prev number not greater(difference) and next number not greater(difference)
                    {
                        //so we know we need to check backwards if less than (within difference)
                        //we do not need to worry about the

                        System.out.println("^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^6");

                        if (!(Math.abs(nums[k] - (nums[k+1] + difference)) <epsilon))
                        {
                            if (!potentialfurtherAscendingBeyondThisStart.equals("")
                            && !(potentialfurtherAscendingBeyondThisEnd.equals("")))
                            {
            sm.add(potentialfurtherAscendingBeyondThisStart+"->"+nums[k]);
            System.out.println("-------------------4444444 STORED TO WRITE RANGE");
            System.out.println("9705Writing range: " + potentialfurtherAscendingBeyondThisStart + "-> " + nums[k]);
            potentialfurtherAscendingBeyondThisStart="";
            potentialfurtherAscendingBeyondThisEnd="";
            System.out.println("CURRENT LIST: " + sm);
                            }
                        }
                        else
                        {

                    System.out.println("2Writing range: " + start + "-> " + end);

                    sm.add(start+"->"+end);
                        }
                    }
```

We know we are in descending area of code since counter!=0
So I have checked if the next is descending.. IF not, then it will terminate the range with current nums [k] value.. Otherwise it will process 2writing range as per usual in else statement



```
[20.0->19.8, 63.5, 14.9, 14.9, 48.6->48.2, 48.2->48.6]
```

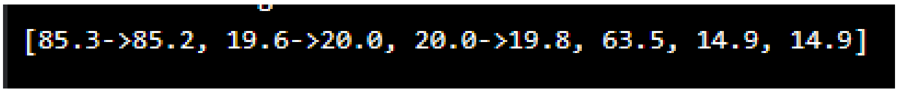
I will now need to go through the ChatGPT data again

Unfortunately as can be seen in the last two rows of the output.xlsx I have landed into all sorts of issues...
I believe these errors were not present before...

So I have rolled back and discarded all the fixes above...
Perhaps I need to put these failed test cases in my code again...
And find a way to implement so it does not violate anything else...

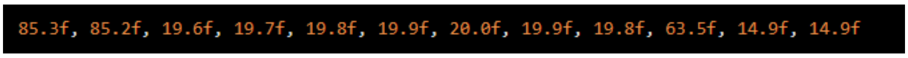
This section of code will be my master test case from now on.. It's a section from ChatGPT data..

There is absolutely no issues with the code last published on my site:
30032025/SummaryRange/8/Final/v1/Solution.java">
85.3f, 85.2f, 19.6f, 19.7f, 19.8f, 19.9f, 20.0f, 19.9f, 19.8f, 63.5f, 14.9f, 14.9f,



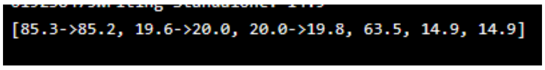
```
[85.3->85.2, 19.6->20.0, 20.0->19.8, 63.5, 14.9, 14.9]
```

I will now try the above test cases in this code... and start process again..



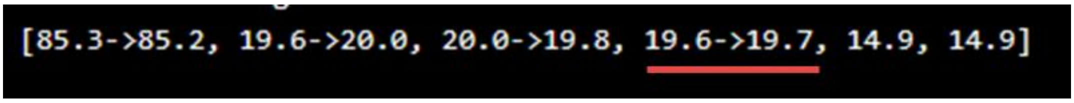
```
85.3f, 85.2f, 19.6f, 19.7f, 19.8f, 19.9f, 20.0f, 19.9f, 19.8f, 63.5f, 14.9f, 14.9f
```

This is the correct state

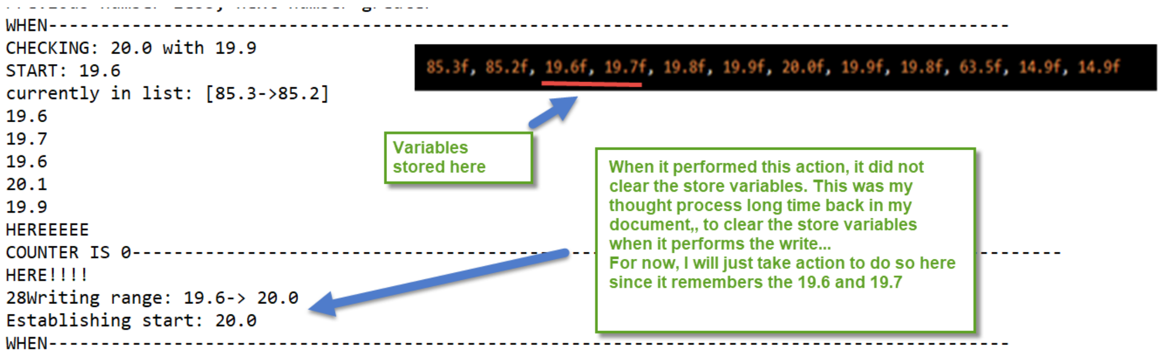


```
[85.3->85.2, 19.6->20.0, 20.0->19.8, 63.5, 14.9, 14.9]
```

And as soon as I apply the fix for error 1, I   finish with:



```
[85.3->85.2, 19.6->20.0, 20.0->19.8, 19.6->19.7, 14.9, 14.9]
```



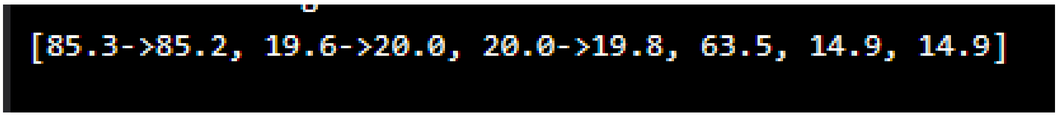
```
WHEN--------------------------------------------------------------------------------------------
CHECKING: 20.0 with 19.9
START: 19.6
currently in list: [85.3->85.2]
19.6
19.7
19.6
20.1
19.9
HEREEEEE
COUNTER IS 0----------------------------------------------------------------------------------------
HERE!!!!
28Writing range: 19.6-> 20.0
Establishing start: 20.0
WHEN--------------------------------------------------------------------------------------------
```

```
85.3f, 85.2f, 19.6f, 19.7f, 19.8f, 19.9f, 20.0f, 19.9f, 19.8f, 63.5f, 14.9f, 14.9f
```

Variables stored here

When it performed this action, it did not clear the store variables. This was my thought process long time back in my document,, to clear the store variables when it performs the write...
For now, I will just take action to do so here since it remembers the 19.6 and 19.7

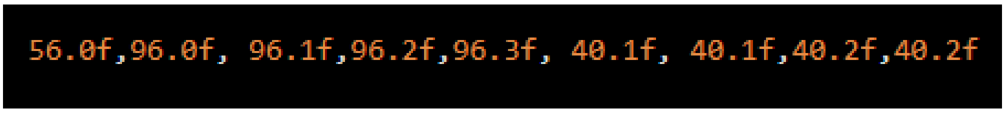
It if fixed again....



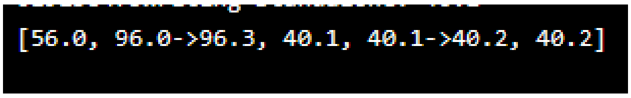
```
[85.3->85.2, 19.6->20.0, 20.0->19.8, 63.5, 14.9, 14.9]
```

I will also make sure the test case associated with error 1 is still functional

TEST CASE 1:



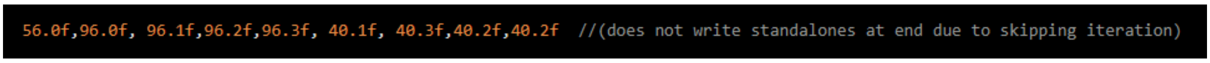
```
56.0f,96.0f, 96.1f,96.2f,96.3f, 40.1f, 40.1f,40.2f,40.2f
```



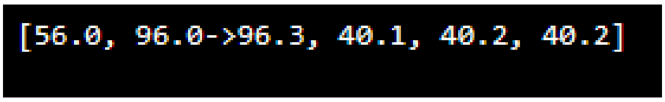
```
[56.0, 96.0->96.3, 40.1, 40.1->40.2, 40.2]
```

All functional

TEST CASE 2:



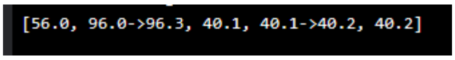
```
56.0f,96.0f, 96.1f,96.2f,96.3f, 40.1f, 40.3f,40.2f,40.2f  //(does not write standalones at end due to skipping iteration)
```



```
[56.0, 96.0->96.3, 40.1, 40.2, 40.2]
```

I have now applied exact same fix as above...
It is beginning to cross my mind since I failed to cleared those store variables it had detriment in lots areas...

And it is fine...

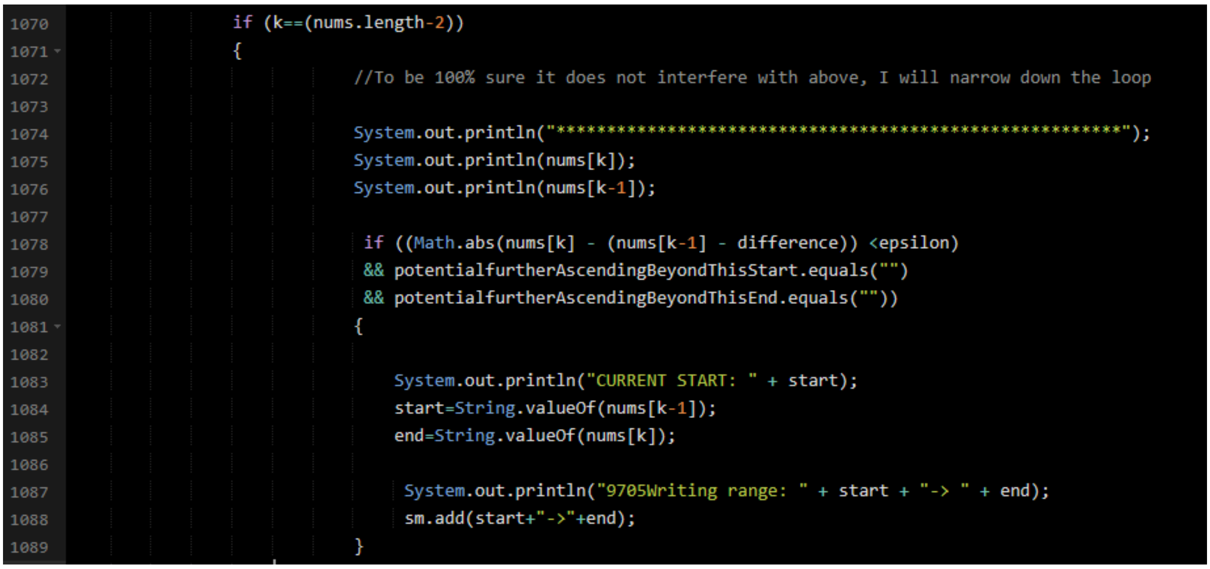


```
[56.0, 96.0->96.3, 40.1, 40.1->40.2, 40.2]
```

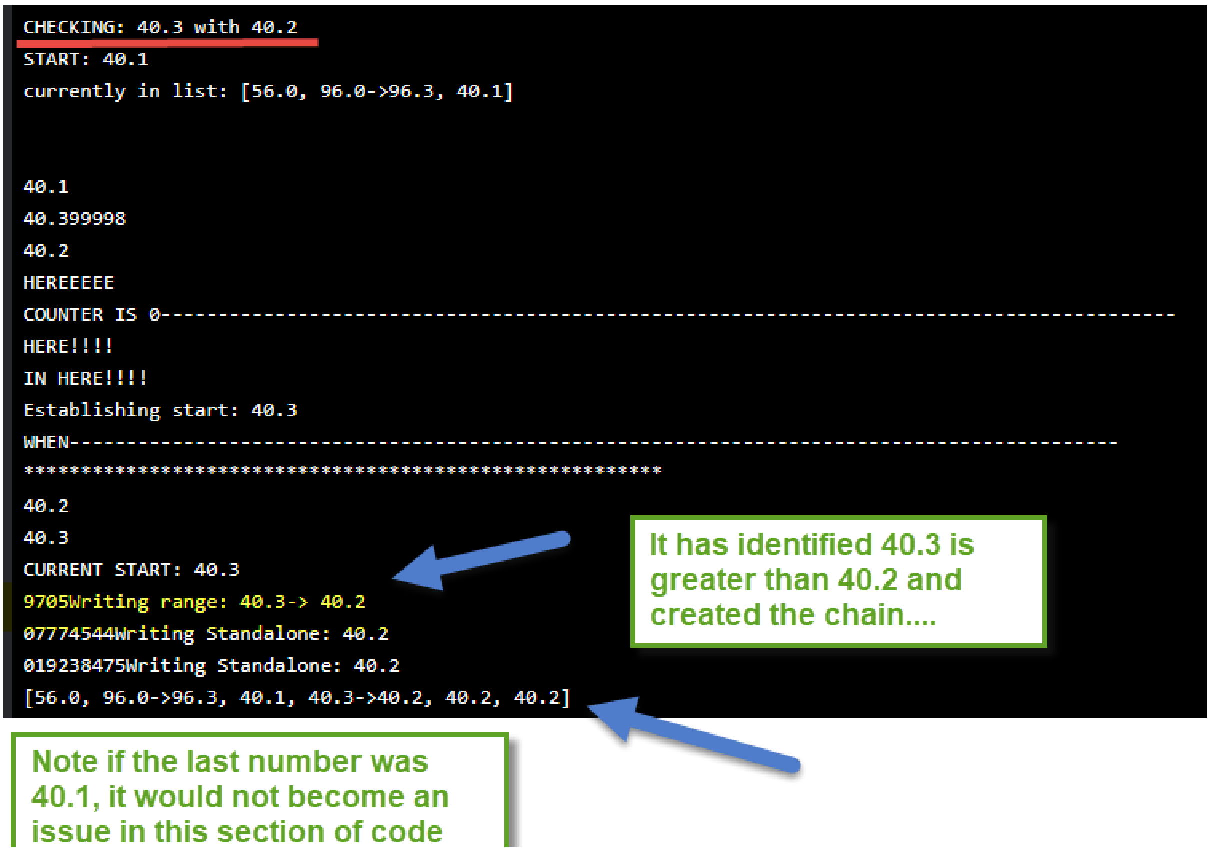
So now, I will try the ChatGPT data again.. I hope this is final effort.
It appears that the fix for error 2 is the cause of the issue...
I will need to try and find another workaround for this problem or try to understand why process is failing ...

I have decided to move the location of the code block (used for error 2) into the else statement here

Reason for this choice was that it still facilitates to check for number before and link it all up...



```
if (k==(nums.length-2))
{
            //To be 100% sure it does not interfere with above, I will narrow down the loop

            System.out.println("********************************************************");
            System.out.println(nums[k]);
            System.out.println(nums[k-1]);

             if ((Math.abs(nums[k] - (nums[k-1] - difference)) <epsilon)
             && potentialfurtherAscendingBeyondThisStart.equals("")
             && potentialfurtherAscendingBeyondThisEnd.equals(""))
            {

                System.out.println("CURRENT START: " + start);
                start=String.valueOf(nums[k-1]);
                end=String.valueOf(nums[k]);

                 System.out.println("9705Writing range: " + start + "-> " + end);
                 sm.add(start+"->"+end);
            }
```



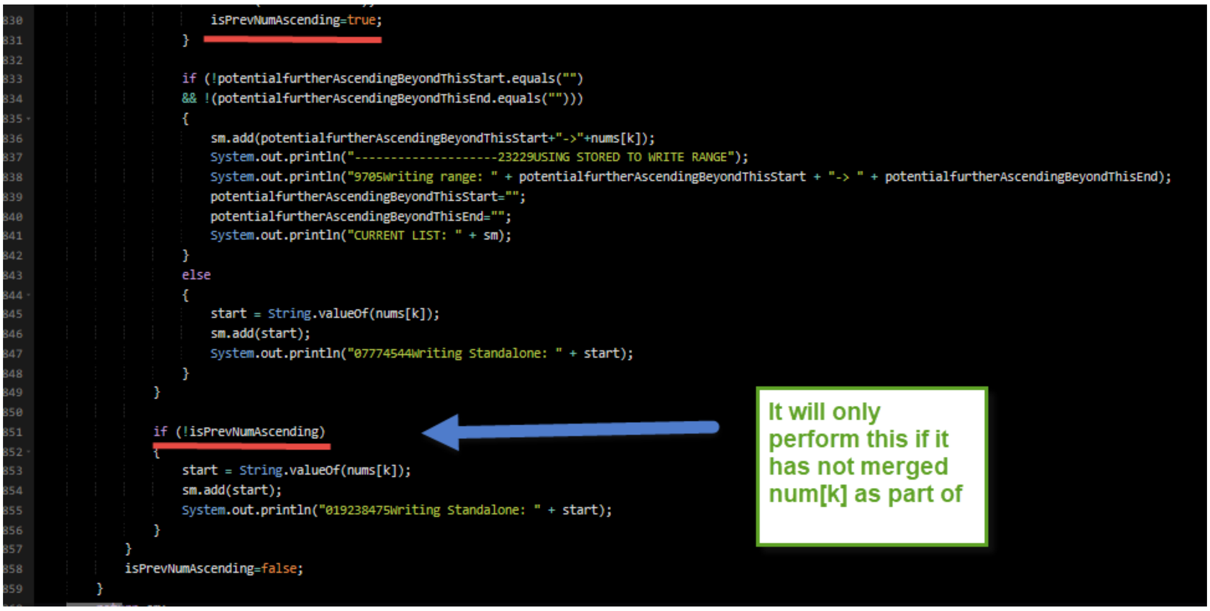
```
CHECKING: 40.3 with 40.2
START: 40.1
currently in list: [56.0, 96.0->96.3, 40.1]


40.1
40.399998
40.2
HEREEEEE
COUNTER IS 0------------------------------------------------------------------------------------------
HERE!!!!
IN HERE!!!!
Establishing start: 40.3
WHEN-----------------------------------------------------------------------------------------------
*******************************************************
40.2
40.3
CURRENT START: 40.3
9705Writing range: 40.3-> 40.2
07774544Writing Standalone: 40.2
019238475Writing Standalone: 40.2
[56.0, 96.0->96.3, 40.1, 40.3->40.2, 40.2, 40.2]
```

It has identified 40.3 is greater than 40.2 and created the chain....

Note if the last number was 40.1, it would not become an issue in this section of code

And now also my ChatGPT code remains ok with no issues.....

But as can be seen above it has written 40.2 twice and also in the merge...
If it is included in the merge, we need to include code to prevent it writing it additional time..



```
                isPrevNumAscending=true;
            }

            if (!potentialfurtherAscendingBeyondThisStart.equals("")
            && !(potentialfurtherAscendingBeyondThisEnd.equals("")))
            {
                sm.add(potentialfurtherAscendingBeyondThisStart+"->"+nums[k]);
                System.out.println("-------------------23229USING STORED TO WRITE RANGE");
                System.out.println("9705Writing range: " + potentialfurtherAscendingBeyondThisStart + "-> " + potentialfurtherAscendingBeyondThisEnd);
                potentialfurtherAscendingBeyondThisStart="";
                potentialfurtherAscendingBeyondThisEnd="";
                System.out.println("CURRENT LIST: " + sm);
            }
            else
            {
                start = String.valueOf(nums[k]);
                sm.add(start);
                System.out.println("07774544Writing Standalone: " + start);
            }
        }

        if (!isPrevNumAscending)
        {
            start = String.valueOf(nums[k]);
            sm.add(start);
            System.out.println("019238475Writing Standalone: " + start);
        }
    }
    isPrevNumAscending=false;
}
```

It will only perform this if it has not merged num[k] as part of